**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
ON APPEAL FROM THE EXAMINER TO THE BOARD
OF PATENT APPEALS AND INTERFERENCES**

In re Application of:	Richard H. Harvey, et al.
Serial No.:	10/648,595
Filing Date:	August 25, 2003
Group Art Unit:	2164
Examiner:	Alicia M. Lewis
Confirmation No.:	4259
Title:	WEB SERVICES APPARATUS AND METHODS

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

APPEAL BRIEF

Appellants have appealed to the Board of Patent Appeals and Interferences ("*Board*") from the Final Office Action dated May 16, 2008 and the Advisory Action dated August 12, 2008. Appellants filed a Notice of Appeal and Pre-Appeal Brief on September 16, 2008 with the statutory fee of $510.00. This Appeal Brief is filed in response to Notice of Panel Decision from Pre-Appeal Brief Review dated November 5, 2008, finally rejecting Claims 1-20.

Real Party In Interest

This Application is currently owned by Computer Associates Think, Inc. as indicated by:

an assignment recorded on 06/21/2004 from inventors Richard H. Harvey and Timothy Bentley to Computer Associates Think, Inc., in the Assignment Records of the PTO at Reel 015485, Frame 0611 (4 pages).

Related Appeals and Interferences

To the knowledge of Appellants' counsel, there are no known interferences or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.

Status of Claims

Claims 1-20 are pending and stand rejected pursuant to a Final Office Action dated May 16, 2008 ("*Final Office Action*"), and a Notice of Panel Decision from Pre-Appeal Brief Review dated November 5, 2008 ("*Panel Decision*"). Specifically, Claims 1-17 and 19 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent Application Publication No. 2006/0059107 issued to Elmore et al. ("*Elmore*") in view of U.S. Patent Application Publication No. 2004/0002955 issued to Gadbois et al. ("*Gadbois*") and U.S. Patent Application Publication No. 2008/0109897 issued to Moran et al. ("*Moran*"). Claims 18 and 20 are rejected under 35 U.S.C. §103(a) as being unpatentable *Elmore* in view *Gadbois* and *Moran* and U.S. Patent Application Publication No. 2003/0236956 issued to Grubbs et al. ("*Grubbs*").

For the reasons discussed below, Appellants respectfully submit that the rejections of Claims 1-20 are improper and should be reversed by the Board. Accordingly, Appellants present Claims 1-20 for Appeal. All pending claims are shown in Appendix A, attached hereto.

Status of Amendments

All amendments submitted by Appellants have been entered by the Examiner.

Summary of Claimed Subject Matter

The present disclosure relates to UDDI Registry and Web Services in general, and in particular to method(s), apparatus and system(s) used in giving practical effect to such services. (Page 1, lines 9-11).

UDDI (Universal Description, Discovery and Integration) is a set of Standards that have been defined to enable applications that use Web Services to quickly, easily and dynamically interact with one another. UDDI is intended to create a platform-independent, open framework for describing services, discovering businesses and integrating system services using the Internet, as well as an operational registry. Refer to the web site www.uddi.org for further details. (Page 1, lines 13-18).

A UDDI registry provides valuable support to systems structured using Web Services. Figure 1a illustrates schematically basic Web Services and UDDI concepts. Figure 1b illustrates schematically a simplified protocol stack for the Web Services environment. UDDI provides a repository for Web Services information and is itself provided by way of a Web Service. (Page 1, lines 19-23).

. . . [I]mplementing the UDDI standards (available at www.uddi.org) on a Directory requires the solving of a number of problem. The UDDI Standards leave many important issues unaddressed, such as:

- The UDDI Standard defines a number of objects, some of which are related by a hierarchy, but UDDI does not define an all-encompassing hierarchy. For example, Business Service objects will come under Business Entity objects, and the Binding Template objects will come under Business Services. Figure 2 illustrates an example of this hierarchy. Business Entity objects are denoted 21, Business Services objects are denoted 22, and Binding Template objects are denoted 23. It is also to be noted that TModel objects, denoted 24, for example, are not hierarchically related to these objects. There are also other concepts such as Publisher Assertions, for example, which are not defined hierarchically.
- creating an efficient implementation of the requirement that a user be permitted to alter only those objects under his/her control,
- creating an efficient implementation that would allow UDDI registries to be distributed,

- creating an efficient implementation which enhances aspects of management and performance of searching and update.
- How to represent complex UDDI objects in a relatively efficient way. For example Business Entity, Business Service, Binding Template and/or TModel have compound repeating elements. In turn these repeating elements could contain further repeating elements. For example, a Business Entity may contain contacts and the contacts may contain addresses. Addresses may contain address lines and phone numbers. Figure 13 illustrates schematically a UDDI concept of a relatively complex object in a Business Entity. The Business Entity object 131, includes, for example, a number of attributes 132, such as AuthorizedName, BusinessKey, and Name. The Name has one or more Name fields 133, such as 'text' or this may implicit in the 'Name' itself. There is also 'language'. There may be one or more of these fields 133.
- How to provide for relatively rapid searching for a specific items contained in repeating elements.
- How to represent UDDI information and requirements in hierarchy of Directory objects.
- How to manage deletion of UDDI objects and all their related information in an efficient manner, and
- How to optimize construction of intermediate search result collections during search operations so that both Directory access and iterative in-memory operations are minimized, taking into account the Directory storage medium limitations. In practice, Directory entries may be stored and returned in arbitrary order, and Directory results may be too large to sort.
- How to represent the data concerning a Publisher Assertion, in an efficient way,
- How to create an efficient implementation of Publisher Assertions, particularly with regard to the implementation of the findrelatedBusiness method,
- How to implement efficient searching of Publisher Assertions by relationship,
- How to manage the validity of a Publisher Assertion,
- How to restrict the assertions created and deleted for a Business Entity are made by the owner of a Business Entity.

- How to efficiently manage related collections of attributes, as defined in the UDDI standard,
- How to define attributes and objects to enhance the performance of searching.

(Page 2, line 20 through Page 4, line 12).

The following summarizes some of the salient features of embodiments of the present disclosure and a few of the advantages provided thereby. (Page 7, lines 19-20.)

According to an embodiment of the present disclosure, a repository layer is created above users so each repository can be placed on a different server. This Repository layer includes one or more Directory nodes which collectively form the Directory pre-fix. This may also be known as 'Domain' or 'Name' of the Repository. An advantage of this is that it provides a single place to hold information about a domain. The name of this node represents the Directory prefix. (Page 7, lines 21-27.)

A user object may be created to hold the data representing a UDDI account. An advantage of this is that it provides a single place to hold information about a user/account. (Page 7, lines 28-30.)

Business Entity object(s) may be arranged under User object(s), Business Service object(s) under Business Entity object(s), and Binding Template object(s) under Business Service object(s). An advantage of this is that a repository or 'domain' layer above the user object layer enables a number of repositories to be posted or logically connected together. The domain layer may be arranged in a number of levels, for example having different countries, AU, US, EP, etc., organized by continent. (Page 7, line 31 - page 8, line 4.)

Another advantage is that this feature may be given effect by use of the Distribution features of an X500 Directory. For example, to implement this, a 'World', or 'Corporation' Node is placed at the top of the virtual Directory tree, and a uniquely named Node is placed at the top of each UDDI sub-tree (UDDI Name Space). While invisible to users, these 'Node' prefixes allow a UDDI repository to leverage Directory distribution. (Page 8, lines 5-10.)

According to an embodiment of the present disclosure, the Business Entity objects can be made a child, of the user object. Having a user/account over the Business Entity, Business Service and Binding Template hierarchy gives the effect of each user having their own sub-tree. This enhances manageability and security. The user is readily restricted to modifying and / or controlling only their own sub-tree. This also enhances performance by making use of Directory sub-tree search operations. (Page 8, lines 11-17.)

According to an embodiment, TModels defined by a user can be made children of the user object, thus makes security easy to implement. This enhances manageability and security since the user can only modify and / or control their own sub-tree. It also enhances performance by making use of Directory sub-tree search operations. (Page 8, lines 18-22.)

An embodiment of the present disclosure represents a 'mapping' of the UDDI environment using X.500/LDAP Directory technology. In particular, the hierarchy structure of the X.500 and LDAP Directory technology has been found to be suitable to the UDDI environment. Careful design of additional elements (such as the user object) have made the hierarchy even more suitable to the needs of the UDDI environment. (Page 8, lines 23-28.)

A distributed registry provides a flexible, scalable solution. In this scenario, each participating office has a separate registry, and each registry views the others as a logical part of its own content. The registry service takes care of all the connectivity details, and customers need not be concerned with geography. (Page 12, lines 3-6.)

The second scenario occurs when an enterprise needs to connect its internal UDDI system to that of a trusted partner, or public Internet registry. In the case of a public registry, in particular, replication is problematic. Internet registry operators may be unwilling to replicate parts of their registry to the enterprise's internal registry. Again, a distributed approach is one answer. At present, there are no UDDI Standards for distribution and the proposals for replication are considered complex. One solution would provide the benefits of a UDDI distributed approach without requiring modifications to the standard. (Page 12, lines 7-14.)

With regard to manageability, as a component performing mission-critical functions within an enterprise, UDDI should meet performance and reliability requirements. It should not just exist as a convenient utility for developers. Read access by clients will be the most frequent and most time-critical usage of a UDDI registry. Performance is optimized for maximum throughput, and the response times of lookup queries should not be affected by more complex searching. Performance should not suffer as the registry grows in size and complexity. The data store underpinning the UDDI Registry should be industrial strength and fully support transactions and automatic recovery. In addition, the UDDI servers should have a high degree of availability and support features such as network fail-over and hot standby. System Administrators should have capabilities to make the UDDI registry easy to maintain, monitor and control. These capabilities include DYNAMIC CONFIGURATION to change controls, rules and settings without taking the service offline, ONLINE BACKUPS AND

TUNING for high availability, ADMINISTRATIVE CONTROLS to stop "trawling" of the registry and prevent denial-of-service attacks, MONITORING via SNMP or other types of alerting mechanisms, AUDITING AND DIAGNOSTICS with separate log files for security, statistics, queries and update information and DEPLOYMENT options to support replication, distribution and routing. (Page 12, lines 15-33.)

Many developer-focused UDDI registries have been introduced. These provide useful capabilities for small development teams, but are not true production quality systems. Web Services deployments are growing rapidly and there is a corresponding need for an Enterprise-quality registry that can scale rapidly to support ongoing Web Service deployments. (Page 13, lines 1-5.)

A UDDI registry provides a service. This service will be relied on by many applications. In the case of on-line businesses, it may be important that this service be ever present. For example, a UDDI registry may be required to provide service level agreements of 99.99% availability. In order to facilitate this level of availability, the UDDI registry may be replicated across two or more machines, and mechanisms provided to make certain that the machines are kept synchronized, and that, should any of the machines become unavailable, any incoming queries are automatically routed to an available machine. (Page 13, lines 6-13.)

As has been pointed out, UDDI may be considered as effectively analogous to phone directory service. As such, the Directory model of information storage is a perfect base on which to build a UDDI registry service. The Directory model has been evolved and developed for the specific needs of Directory-based services, with the security, scalability and reliability needed for enterprise level deployment. (Page 13, lines 14-19.)

Most of the items described above are implemented at the service level, rather than at the data storage level, in application architecture. Relational databases (RDBMS) are generic toolkits upon which many different kinds of applications can be built. RDBMS implementations concentrate on providing solid data access functionality rather than extra service functions that are required in the end application. (Page 13, lines 20-25.)

The Directory Service architecture shown in Figure 3 illustrates the separation of a Service Layer 31 from the other components. Encapsulating the interface functions into a Service Layer 31 results in reusable service infrastructure components. An excellent example of this is a web server. A web server provides a collection of functions (HTTP access, CGI processing and so on) that together make up a service useful enough to build into a standalone component. In the same way, the Directory Service model has been developed to supply the functions required by a specific type of application. Directory technologies provide the underpinning for many mission-critical applications in the area of authentication and authorization. (Page 13, line 26 - page 14, line 2.)

UDDI may be viewed as analogous to another kind of Directory Service. It may then be seen that many of the implementation problems posed by UDDI can be solved by using Directory technologies. For example, Directories are optimized for extremely efficient find and search operations that are very common for UDDI phone Directory operations. (Page 14, lines 3-7.)

It has already been noted that a UDDI service should offer strong security, distribution and manageability capabilities if it is to be deployed successfully in the Enterprise. These are the very same attributes which have already been built into Enterprise-strength Directory Services solutions. (Page 14, lines 8-11.)

One way to construct an Enterprise UDDI registry is to extend the existing Directory infrastructure, which has been tried and tested in high-performance, real-world applications. (Page 14, lines 12-14.)

The Directory Services architecture provides the optimal vehicle to implement an Enterprise UDDI registry. This combination supports the capabilities necessary for success. The UDDI Service as illustrated schematically in Figure 4 identifies components which may be implemented for this infrastructure. UDDI SEMANTIC BRIDGE 41 is a service component that mediates between the SOAP implementation 42 of UDDI and the LADP interface 43 supported by Directory 44. Directory 44 delivers information access with supporting security controls, distribution mechanisms, and administration capabilities. RDBMS 45 provides the underlying physical data management, transactional integrity and backup and recovery mechanisms. (Page 14, lines 15-24.)

UDDI registry products may be built directly on RDBMS technology. Relational Databases, although very useful and strong in many ways, do not by themselves meet the requirements unique to Directory processing. (Page 14, lines 25-27.)

It would be possible to build a Directory-type application from scratch, utilizing an RDBMS or other data storage system underneath. However, this may not be the most efficient approach. (Page 14, lines 28-30.)

An alternative approach is to apply the Directory Service model to deliver a UDDI registry and supply the functions required for this specific type of application. Even more functions required for a UDDI registry could be supplied by modern, industrial-strength Directory Services. A UDDI registry may be viewed as a Directory Service with specialized communications and APIs. Delivering UDDI services on a Directory could provide the requisite security, distribution and management capabilities without having to modify the UDDI Standards to gain the benefits. (Page 14, line 31 - page 15, line 5.)

A careful design of the data representation would be beneficial to give the functionality and performance required of a UDDI repository. (Page 15, lines 6-7.)

The following description refers to various UDDI concepts. A more detailed description of these UDDI concepts can be gained by reference to the UDDI specifications (http://www.uddi.org/specification.html). (Page 15, lines 8-10.)

A schema, in Directory parlance, is a description of the data elements that can be stored in the Directory, and how those elements may be connected together. This includes descriptions of each of the possible attributes (an attribute holds a single piece of data), descriptions of the various objects (an object is a collection of attributes), and specifications of the possible object hierarchies. The particular Schema notation used in this specification is the one used by eTrust Directory, a product of Computer Associates International Inc. 'eTrust' is a product name and trademark of Computer Associates International Inc. Of course, other Schema notations my be used. (Page 15, lines 11-19.)

The present disclosure describes a Schema used to implement a UDDI repository using a Directory as the data store. There are a number of concepts involved in this Schema. There are also a number of techniques used to enhance the operation of the UDDI implementation. The following is a brief description of some of these concepts. A more detailed description of these concepts and techniques will be described later below when describing embodiments of the present disclosure. (Page 15, lines 20-26.)

The present Schema is designed to provide optimized operation. The present Schema design, which includes the definition of Attributes, Object Classes, Entries and the Hierarchy, is embodied in a manner that enhances operation. The present Schema design provides significant advantages in, at least, security, performance, manageability, and distribution. (Page 15, lines 27-31.)

The hierarchy of the system will now be described. An X.500 Directory supports distribution internally, providing a distributed UDDI repository without any coding at the UDDI level. A level divides the contents of the repository. The (optional) domain level of this schema provides that level, each domain entry, and all of the entries below it, can be placed on a separate Directory server transparently to the UDDI-level programming. Figure 11 illustrates an embodiment of this aspect of the present disclosure. This will be described in more detail later below. (Page 15, line 32 - page 16, line 6.)

According to an embodiment of the present disclosure, a user object is placed over the business and TModel objects. The user object provides a place for the storage of information relating to the user. It also provides an anchor point for all of the data published by the user. Figure 10 illustrates an embodiment of this aspect of the present disclosure. This will be described in more detail later below. (Page 16, lines 7-12.)

Security is facilitated in this domain / user hierarchical system. A UDDI implementation can enforce that a user has control over their sub-tree of data objects. (Page 16, lines 13-15.)

Searching for user controlled entries is provided. Searching for data controlled by this user can be enhanced by using a sub-tree search under the user object. (Page 16, lines 16-18.)

It is possible to find a business by specifying, for example, a TModel that occurs in a Binding Template. This equates to "finding x by finding one (or more) of its children". In other words, a query may be "find all businesses which have a service which has a Binding Template which references this TModel". Such queries are done by finding the DN (Distinguished Name) of the descendent object, and discarding the unwanted levels, to yield the DN of the Business Entity. It is also possible to do duplicate elimination in this manner. This find feature comes about due to the hierarchical nature of the structure of the present disclosure. (Page 16, lines 19-27.)

Searching may be performed using attributes unique to an object class. This is an optimization that has two advantages. This simplifies the writing of searches, and yields superior performance through the elimination of 'weak' clauses. A 'weak' clause is a part of a filter that returns a large number of entries, or which refers to an attribute that is part of many entries. A design which used the same attribute name for every Name would have two choices when searching, for a Business Entity by name: it includes the object class in the search or filter the results of the search. The former is only possible if business names had a unique object class, and even so, object class is a weak clause, incurring more overhead. The latter means extra code and the potential for returning a result list much larger than the desired result. (Page 16, line 28 - page 17, line 5.)

For example, consider a company called “McKenna’s Testing Services” which offers a wide range of Web Services, all of which include “McKenna’s” in their name — a search for business entities with “McKenna’s” in their name would return intermediate results for all of the services as well. These intermediate results may be eliminated, but dealing with them reduces performance. (Page 17, lines 6-10.)

It is preferable to be able to specify an attribute name in a search and have that attribute name uniquely, identify the object class being sought. To continue the example above, the search is much simpler if we can specify: (euBusinessEntityName = McKenna’s*) (Page 17, lines 11-14.)

Such a design produces strong searches, which are efficient because they are searching only the desired area. Strong searches include searches which return a small number of entries. The Directory can index the euBusinessEntityName attribute, and return results from that index — this produces good performance, and avoids handling unnecessary intermediate results. (Page 17, lines 15-20.)

For simple queries, such a design means that a search for a Business Entity name is a single clause, rather than the compound that might be necessary in another design. Imagine if the name attribute were called euName, and the Business Entity name object were called euBusinessEntityName. That would yield a search like:
(&(euName = McKenna’s*)(oc=euBusinessEntityName)) (Page 17, lines 21-26.)

There is an even more simple design, wherein all names are stored in the same object class. This means that the search reduces to (euName=McKenna’s*) again, but now we wade through results for all names, trying to locate those which have a parent object that is a Business Entity — this last design would yield potentially poor performance, and rather more complex programming. (Page 17, lines 27-32.)

Shadow attributes may be used for case-sensitivity. It is far from trivial to provide both case-sensitive and case-insensitive searching using a single index. One option is to index case-insensitively, then scan the results case-sensitively. Another solution here is to index the original data case-sensitively, and to add a second attribute (in which the same data is stored) which is indexed case-insensitively. Then all that is required is to choose the appropriate attribute to search depending on the find qualifiers. (Page 18, lines 1-7.)

Every attribute in this design may be single-valued. This allows efficient indexing, higher performance, and stronger searches. (Page 18, lines 8-9.)

Using multi-valued attributes makes ambiguous searches possible. That is, it is possible to get search results which are counter-intuitive, and unintended. Imagine a multi-valued numeric attribute, called 'n', and an entry which contains this attribute with the values 2 and 5; this entry will be returned in response to a search (&(n < 3)(n > 4)), which is not something that would be readily anticipated. (Page 18, lines 10-14.)

Single-valued attributes are one of the techniques used for strong searches. A strong search is one which can eliminate the majority of candidate results through the index. Strong searches are a key to improved performance. (Page 18, lines 15-17.)

Aliases may be used for service projection. This is a significant benefit of using an X.500 Directory as the data store. A service projection can be represented neatly using an X.500 alias. This has the major advantage of guaranteeing data integrity. The alias accesses the original data, so any change to the original is instantly reflected by the alias. If the Directory implementation supports alias integrity, then when the original entry is deleted the alias vanishes without additional work. (Page 18, lines 18-24.)

Publisher Assertions are one of the least clearly defined elements in the UDDI Standard, and they require careful design. An inappropriate implementation could readily yield poor performance. (Page 18, lines 25-27.)

Because the most common use of Publisher Assertions is the find_relatedBusiness API, which is searching for all the completed Publisher Assertions relating to a specified Business Entity, it is good design to place each assertion under a Business Entity to which it refers. (Page 18, lines 28-31.)

By calculating the status of the assertion, and storing it in the assertion object, it is possible to restrict a search to completed Publisher Assertions. This means that the results returned will not contain spurious references that are to be removed. (Page 18, line 32 - page 19, line 2.)

Storing the relationship object as an auxiliary class allows the search to eliminate any assertion which has an unwanted relationship. If the relationship were stored as a child object, it would not be possible to write a single search that would address both the relationship and the assertion completion status. (Page 19, lines 3-6.)

UDDI keys may be used for naming where present. UDDI defines keys for many of the important object classes, and these keys are specified as being guaranteed to be unique. This means that the keys can be used as the naming attributes for the objects. Using the UDDI keys as the naming attributes means that there is no need to attempt resolution of naming clashes – that would be required if, for example, the default name were used as the naming attribute for a Business Entity. (Page 19, lines 7-13.)

Keys may be provided for naming where not present. That is, not all UDDI objects have defined keys. An example is Publisher Assertions. For these, the present system provides a key, using the same algorithm as is used for the UDDI-defined keys. This re-use of the idea means that code and structure written for the other objects can be re-used. (Page 19, lines 14-18.)

Where a series of UDDI objects are children of another object, and the order of the children is important (address lines, for example), the keys assigned to the child objects are arranged to be monotonically increasing in value, so that sorting on the keys yields the desired order. This simplifies the process of ensuring the desired order. (Page 19, lines 19-23.)

Where practical, it is desirable that keys vary in a little-endian manner. That is, the leftmost byte of the key varies most rapidly, because that yields the best performance of indexing in the X.500 Directory being used as the data store. (Page 19, lines 24-26.)

The UDDI Standards define a number of sub-structures inside some of the main object types. In many cases these sub-structures are optional, and may be repeated (they may occur zero, one, or more than one times in the same object). A simple example is the name sub-structure, containing a string (the name) and a language identifier. The X.500 schema definition does not support the use of structured attributes, so there is no immediately clear mapping of sub-structures. There are a few ways in which these sub-structures can be implemented in an X.500 schema. (Page 19, line 27 - page 20, line 2.)

One way is to concatenate the components of the sub-structure into a single attribute, using a separator of some kind to divide the various elements. This may not be the optimum design choice, because it loses the ability to index or search the components separately, and it adds processing complications to handling the data. (Page 20, lines 3-7.)

In the present system, the particular design used to represent substructures is chosen to maximise performance and manageability. The design disclosed may use one or more of a variety of techniques to represent sub-structures in a Directory. These techniques can be summarized in 3 categories. (Page 20, lines 8-11.)

One technique is that, many of the sub-structures can be handled as child objects. Names are a good example: the Business Entity names are stored as children of the Business Entity. Another example is descriptions, where a separate Business Description object is a child of Business Entity objects. Figure 15 provides an illustration of an embodiment of this aspect of the present disclosure and will be described in more detail below. (Page 20, lines 12-17.)

Another technique is flattening/merging. In cases where there may be at most one relationship to another object, the attributes may be combined into a single object. In this case, the hierarchy is said to be flattened because two objects have been combined into one object. A new object is said to be merged because the new object contains a combination of attributes from the combining objects. Preferably, the contents of the Relationship Object are promoted to the Parent Object. (Page 20, lines 18-24.)

For example, Figure 16 illustrates schematically a representation of a UDDI relationship diagram Figure 17 illustrates schematically a Directory Hierarchy diagram where the Directory hierarchy has been formed by a flattening of the UDDI objects. (Page 20, lines 25-28.)

By way of explanation, Figure 16 illustrates Object 161 having a relationship Object 162 to Object 163. (Page 20, lines 29-30.)

In accordance with an embodiment of the present disclosure, where there is a one-to-one relationship, a 'child' can be promoted. In other words, that part of the hierarchy can be collapsed or flattened and objects merged. The result is illustrated schematically in Figure 17. The Parent Object 171 has contents A1, A2, An and has one or more children, Child Object 9n, with contents B1, B2, Bn, C1, C2 and Cn. (Page 20, line 31 - page 21, line 3.)

Another technique is splitting. For example, in one particular case (the OverviewDoc sub-structure), a sub-structure contains an unrepeated element and a repeated element. The unrepeated element (OverviewURL) can be moved into the parent, while the repeated element can be made a child object. (Page 21, lines 4-7.)

Another aspect of the present disclosure is management. Deleting a TModel hides it from find_TModel but does not remove it from the repository. Accordingly, to implement the correct handling of TModels, a hidden flag may be implemented. The presence of this flag indicates that a TModel (or user object) is hidden. The absence of the flag indicates that it is not. This will be the case for the vast majority of TModels, so this approach is efficient. No space is occupied in unhidden objects, and no indexing is used either. The Directory will index only those entries which do have the hidden attribute. This also means that the search for unhidden TModels will be fast and efficient. (Page 21, lines 8-16.)

The X.500 Directory used as a data store encourages a design which does not store empty values. For example, a (optional) value which is absent from the object is not stored in the Directory. This makes efficient use of storage space, and makes for stronger searches. Any search on an attribute need only consider those objects which have data for that attribute. (Page 21, lines 17-21.)

The data hierarchy of the present system matches well with the intent of the UDDI Standard. When a request arrives to delete a UDDI object, it maps directly to the deletion of a sub-tree in the Directory. For example, deleting a service includes deleting its names and descriptions, and all of its Binding Templates. All of these are children of the service entry in the Directory. Accordingly, the present system deletes the sub-tree from the service entry on down. This is readily implemented, and efficient. (Page 21, lines 22-28.)

A domain is a name representing the base of a hierarchical sub-tree. In X.500 terminology a domain is known as a context prefix. In LDAP terminology it is known as a suffix. Giving UDDI repositories a domain name allows use of true distribution (in the X.500 sense) of the data in the repository. The UDDI Standard only supports replication. By having the domain nodes, the present system can use Directory distribution facilities transparently to the application. (Page 21, line 29 - page 22, line 2.)

For example, assume that an Enterprise deploys UDDI internally, but has two development sites. With this facility, they can deploy a UDDI server at each site, with distribution allowing each site to transparently view items published on both registries. (Page 22, lines 3-6.)

An advantage of this is that it allows distribution ‘for free’. For example, the UDDI server does not have to do any extra work and the Directory system effectively links together islands of information. (Page 22, lines 7-9.)

Nothing in the UDDI Standards dictates how the user information is stored. By creating user objects, all of the information relating to a user can be stored in a single object, and that object, can be used as the root of the sub-tree holding all of the objects that the user publishes. This makes the definition of security much simpler. For example, if the object under consideration (be it business, service, or even TModel) is underneath the user's user object, then the user controls it. (Page 22, lines 10-15.)

UDDI defines objects that contain repeating elements. For benefits such as performance, searchability and manageability these repeating elements can be represented as child objects. (Page 22, lines 16-18.)

Storing repeating structured data as child objects allows representation of the data efficiently in a Directory, with each field individually available (and indexed) for searching. (Page 22, lines 19-21.)

For example, Business Entity names can be stored as children of the Business Entity object. Another example is Business Description which can be stored as children below Business Entity objects. (Page 22, lines 22-24.)

An advantage of this type of system is that it allows searching for a name (which is a common UDDI search), and the DN of the entry gives the DN of the object to which the name belongs. (Page 22, lines 25-27.)

UDDI defines redundant 'container' nodes (UDDI structures which contain only child sub-structures, rather than attributes). These can be removed because they can be constructed at relatively low cost from the results of a query. In some cases, attributes can be promoted from a child node to its parent, to remove the now-redundant child-node from the Directory representation. (Page 22, lines 28-32.)

For example, tModelInstanceDetails is not represented in the Directory schema as it contains no attributes. instanceDetails is not represented in the Directory schema as its attributes were promoted into the tModelInstanceInfo parent, 'as were the attributes of its child, overviewDoc. The category and identifier bags are not represented in the Directory, their contents are made children of the owner of the bag. (Page 23, lines 1-6.)

An advantage of this is that it reduces the number of entries in the Directory. In particular, it minimizes the depth of the DIT, which can improve performance. (Page 23, lines 7-9.)

Figure 12 illustrates schematically a hierarchy structure according to an embodiment of the present disclosure. One or more Domain or Prefix 121 are provided. Under each Domain 121, there may be one or more Users 122. Under each User 122, there may be provided one or more TModel 123 and one or more Business Entity (BE) 124. Under each Business Entity 124, there may be provided one or more Publisher Assertion (PA) 125, one or more Business Service (BS) 126 and one or more Service Projection (SP) 127. Under each Business Service (BS) 126, there may be provided one or more Binding Template (BT) 128. Aliases can be placed as required by a particular implementation. For example, Service Projection object(s) (shown as a triangle in Fig. 12) may stem as an alias from Business Entity object(s). (Page 23, lines 10-20.)

The advantages of this schema design as represented in Figure 12 will become apparent from a reading of the present disclosure as a whole. (Page 23, lines 21-22.)

Publisher Assertions are placed under the business entities to which they refer because they are most frequently used in the context of a find_RelatedBusinesses call, which specifies a business key and is looking for all the businesses related to that one via Publisher Assertions. The present system locates the specified business, then reads all the Publisher Assertions underneath it (that are complete). This is a quick and efficient way of locating the relevant assertions. (Page 23, lines 23-29.)

An advantage of this is that it allows fast and efficient searches. It also allows easy maintenance of data integrity. For example, when a business is deleted, any Publisher Assertions are automatically deleted too. (Page 23, lines 30-32.)

TModels can be changed (or retired / hidden) by the user who published them. Placing them under the entry representing the user makes the security simple. For example, if the TModel lies in the sub-tree under the user entry, then it can be modified. If not, then it can not. (Page 24, lines 1-4.)

In more detail, if the DN (Distinguished Name) of the user trying to make the change matches a prefix of the DN of the TModel, the entry can be modified by that user, otherwise it can not. The Directory can be used to make this determination (Naming exception if the DN doesn't exist), or the UDDI server can do it. (Page 24, lines 5-9.)

When an object is deleted from the repository, the information associated with that object may also be deleted. This is greatly simplified by the hierarchical design used according to embodiments of the present schema. When the object is deleted, the entire sub-tree of which it is the root can be deleted, and this process can delete all (and generally only) the associated information. Deleting a sub-tree can be performed bottom-up. Each entry can only be deleted when all its children are deleted. This is managed by listing all the children in reverse DN order. This guarantees deletion of the children before their parents. (Page 24, lines 10-17.)

An advantage of this is that a sorted list method is an alternative to the more complex use of recursion. Further, it is relatively simple and memory-efficient. When all the entries in the subtree are sorted by DN, and deletes are executed in reverse order, this guarantees that all children will be deleted before their parent. (Page 24, lines 18-22.)

For example, when a business service is deleted, the system deletes all the Binding Templates associated with it, their TModel instance information, and the various associated category information. All this can be deleted by deleting the sub-tree of which the business service is the root. (Page 24, lines 23-26.)

Due to the hierarchy used in the design of this schema, the DN of an object reveals the chain of ownership and control for an object. Note that inference is also dependent on careful choice of naming attributes. (Page 24, lines 27-29.)

An advantage of this is that it can reduce the number of searches or reads used to gather information. For instance, with search results which are child objects (such as names), the DN of each entry reveals the parent (e.g. the BusinessEntity) and the owning account. (Page 24, lines 30-33.)

For example, the DN of a business service reveals the business to which it belongs, and the user who controls it. (Page 25, lines 1-2.)

Directories do not guarantee any ordering of the result. When dealing with a complex result (such as a Business Entity and its business services, together with their appropriate names and descriptions), the construction of the output can be simplified by taking the results of the search and sorting them by DN. This organizes them so that the construction of the results becomes relatively simple. Each object is constructed before its children, so it is easy to place the children under their parent, so that the result for a business is organized before its services. All the children of an object appear before the next object of the same type, all of the services for one business before the next business appears. This also allows simple recursive construction, because the same thing applies at each level. (Page 25, lines 3-13.)

An advantage of this is that it minimizes the number of passes through a list of raw entries required to construct the UDDI structures. (Page 25, lines 14-15.)

For example, after sorting, the result for a business, A, is followed by a result for its first service, AA, that service's name, then A's second service, AB, and its names, then a second business, B. (Page 25, lines 16-18.)

A search can also be carried out on children. For example, a frequent search request may be "finding x by finding one (or more) of its children". One of the ways a business can be found by a search is by specifying, for example, a TModel that occurs in a binding template. In other words, the query is "find all businesses which have a service which has a binding template which references this TModel". These queries can be done by finding the DN of the descendent object, and chopping off the unwanted levels to yield the DN of the business entity. Advantageously, this also eliminates duplication. This search method comes about, in part, due to the hierarchy structure of embodiments of the present disclosure. (Page 25, lines 19-28.)

The use of guaranteed unique keys simplifies matters. The entire repository can be searched for a single key, and uniqueness will assure that there will either be no result (if that key is not present), or one result (if it is present). There is no need to be cautious about limiting searches within the range of a parent. This yields enhanced performance from the Directory, because it can use database indexes to their optimum. (Page 25, line 29 - page 26, line 2.)

An advantage of this is that it makes use of the fastest type of Directory queries. Another advantage is that the guaranteed unique names may be important if a given object is referenced from another. (Page 26, lines 3-5.)

A property of most indexing systems is that they are data dependent. If the data is "little endian" (the leftmost portion changes most rapidly) that data tends to be spread and so the indexes can give maximum performance. Conversely, if the data is repetitious, the indexes may not be very effective. A UUID (Universally Unique Identifier) algorithm can be used which exhibits "little endian" qualities. An advantage of this is that it maximises Directory performance. (Page 26, lines 6-11.)

Keys may be added to derived objects. Where a repeating data element is made into a child object, there is a need to add a naming attribute, which will form the last arc of its DN. In a Directory, the naming attribute is different from its siblings, since no two children of the same parent can have the same name. (Page 26, lines 12-15.)

Two kinds of keys may be used. For child objects which do not require order, UUIDs are used because these are guaranteed to be unique. Where order is important, keys with a monotonically increasing property are used to guarantee order. (Page 26, lines 16-19.)

In the UDDI Standard, a Business Entity can offer two kinds of services: those which it controls (represented in the repository by child objects), and those which it offers an interface to, despite the fact that they are provided by another Business Entity. The latter are represented in the disclosed UDDI repository by aliases. An alias provides exactly the right features. For example, if the original object (service) is altered in some way by its owner (perhaps another Binding Template is added), then the object referenced via the alias "changes" too. Moreover, any search under the Business Entity for a service will yield both real and aliased services. (Page 26, lines 20-28.)

For example, aliases can be used for Service Projection, where a Business can point to a Service defined under another Business. (Page 26, lines 29-30.)

An advantage of this is that leveraging aliases allows functionality that basically involves "an alternative name" to be automatically provided. Furthermore, if the Directory supports alias integrity, then if the original Service is deleted, any projections are automatically removed. (Page 26, line 31 - page 27, line 2.)

In the UDDI Standard there are a number of places in which we do not wish to have direct reference to another object, but rather an intermediate step - such as in the case of TModel instance information, or the references to business entities in a Publisher Assertion. In these cases, an alias would complicate the code. Accordingly, instead the present system may use a reference to the object. Because the present system, according to an embodiment, guarantees that every object has a unique key, then that key behaves exactly as a reference, sometimes known as a "foreign" key. (Page 27, lines 3-10.)

Attribute grouping can be performed using auxiliary object class. In handling Publisher Assertions there is a need for an ability to locate a Publisher Assertion using those three attributes which uniquely identify the Publisher Assertion: the two Business Entity keys, and the relationship between them. However, the relationship is specified as a keyed reference, which is itself three different attributes: TModel key, key name, and key value. One way is to store this relationship as a child object of the Publisher Assertion. However, this may not allow the most efficient search for a specific Publisher Assertion. By making the relationship keyed reference an auxiliary class to the Publisher Assertion entry it is possible to search for all five attributes in a single search, and thus isolate exactly the Publisher Assertion objects required. (Page 27, lines 11-21.)

One design of this schema may use normal object-oriented design techniques, and yield, for example, all keyed references having the same attribute names. However, this design may make it more difficult and expensive to isolate, for example, a Business Entity category keyed reference, and to avoid confusing it with a TModel category keyed reference. It may also make it necessary to include object class terms in the filter and such terms are weak (highly repetitious in the repository). (Page 27, lines 22-28.)

Giving, for example, every different kind of keyed reference a different object class and different attribute names, means that any search for a particular attribute name necessarily implies the object class. It also means that the Directory server can construct an index that only has entries in it for the specific kind of entry desired. Such an index will be smaller and consequently faster. (Page 27, lines 29-33.)

For example, a search like: "euBusinessEntityName=Smith*" will consult the index for euBusinessEntityName, and so cannot be confused by an entry containing Smith in an attribute called euTModelName. (Page 28, lines 1-3.)

With regard to the independent claims currently under Appeal, Appellants provide the following concise explanation of the subject matter recited in the claim elements. For brevity, Appellants do not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements. Additionally, this explanation should not be used to limit Appellants' claims but is intended to assist the Board in considering the Appeal of this Application.

For example, Claim 1 recites:

> A method for use in a Web Services arrangement (e.g., Figure 12, reference numerals 121-128; Page 21, line 17 through Page 25, line 2) comprising:
>
> arranging User object(s) under a repository layer comprising one or more Repository objects collectively forming a Prefix, each User object representing a Web Services account (e.g., Figure 5; Figure 12, reference numerals 121 and 122; Page 21, line 29 through Page 22, line 2; Page 23, lines 10-20);
>
> arranging Business Entity object(s) under User object(s) (e.g., Figure 12, reference numerals 122 and 124, Page 23, lines 10-20);
>
> arranging corresponding TModel object(s) under at least one of User object(s), Repository object and Prefix (e.g., Figure 6; Figure 12, reference numerals 121-123; Page 8, lines 18-22; Page 24, lines 1-4);
>
> receiving a request to modify an object from a user (e.g., Figure 12,a reference numerals 121-128; Page 24, lines 1-17; Page 24, line 27 through Page 25, line 2);
>
> matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object (e.g., Figure 12, reference numerals 121-128; Page 24, lines 5-9);

providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user (e.g., Figure 12, reference numerals 121-128; Page 24, lines 5-9); and

modifying the object as requested by the user in response to the user accessing the object (e.g., Figure 12,a reference numerals 121-128; Page 24, lines 1-17; Page 24, line 27 through Page 25, line 2).

As another example, Claim 8 recites:

A computer recording medium including computer executable code for use in a Web Services arrangement (e.g., Figure 18, reference numeral 180; Page 7, lines 14-18) comprising:

code for arranging User object(s) under a repository layer comprising one or more Repository objects collectively forming a Prefix, each User object representing a Web Services account (e.g., Figure 5; Figure 12, reference numerals 121 and 122; Page 21, line 29 through Page 22, line 2; Page 23, lines 10-20);

code for arranging Business Entity object(s) under User object(s) (e.g., Figure 12, reference numerals 122 and 124, Page 23, lines 10-20);

code for arranging corresponding TModel object(s) under at least one of User object(s), Repository object and Prefix (e.g., Figure 6; Figure 12, reference numerals 121-123; Page 8, lines 18-22; Page 24, lines 1-4);

code for receiving a request to modify an object from a user (e.g., Figure 12,a reference numerals 121-128; Page 24, lines 1-17; Page 24, line 27 through Page 25, line 2);

code for matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object (e.g., Figure 12, reference numerals 121-128; Page 24, lines 5-9);

code for providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user (e.g., Figure 12, reference numerals 121-128; Page 24, lines 5-9); and

code for modifying the object as requested by the user in response to the user accessing the object (e.g., Figure 12,a reference numerals 121-128; Page 24, lines 1-17; Page 24, line 27 through Page 25, line 2).

As still another example, dependent Claim 7 recites:

providing a Distinguished Name of an object revealing a chain of ownership and control for the object (e.g., Figure 12, reference numerals 121-128; Page 24, lines 5-9; Page 24, line 27 through Page 25, line 2).

Claim 14 recites certain analogous operations.

Grounds of Rejection to be Reviewed on Appeal

Are independent Claims 1 and 8 (together with Claims 2-6 and 17-18 that depend on Claim 1 and Claims 9-13 and 15-20 that depend on Claim 8) unpatentable under 35 U.S.C. §103(a) over U.S. Patent Application Publication No. 2006/0059107 issued to Elmore et al. ("*Elmore*") in view of U.S. Patent Application Publication No. 2004/0002955 issued to Gadbois et al. ("*Gadbois*"), and further in view of U.S. Patent Application Publication No. 2008/0109897 issued to Moran et al. ("*Moran*")?

Are dependent Claims 7 and 14 unpatentable under 35 U.S.C. §103(a) over *Elmore* in view of *Gadbois*, and further in view of *Moran*?

Arguments

Claims 1-17 and 19 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent Application Publication No. 2006/0059107 issued to Elmore et al. ("*Elmore*") in view of U.S. Patent Application Publication No. 2004/0002955 issued to Gadbois et al. ("*Gadbois*"), and further in view of U.S. Patent Application Publication No. 2008/0109897 issued to Moran et al. ("*Moran*"). Claims 18 and 20 are rejected under 35 U.S.C. §103(a) as being unpatentable *Elmore* in view *Gadbois* and *Moran* as applied to Claims 1-17 and 19 above, and further in view of U.S. Patent Application Publication No. 2003/0236956 issued to Grubbs et al. ("*Grubbs*"). For at least the following reasons, Appellants respectfully submit that these rejections are improper and should be reversed by the Board. Appellants address independent Claims 1 and 8 and dependent Claims 7 and 14 below.

I. Legal Standard for Obviousness

The question raised under 35 U.S.C. § 103 is whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art at the time of the invention. One of the three basic criteria that must be established by an Examiner to establish a *prima facie* case of obviousness is that "the prior art reference (or references when combined) must teach or suggest ***all the claim limitations***." *See* M.P.E.P. § 706.02(j) citing *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991) (emphasis added). "***All words*** in a claim must be considered in judging the patentability of that claim against the prior art." *See* M.P.E.P. § 2143.03 citing *In re Wilson*, 424 F.2d 1382, 1385 165 U.S.P.Q. 494, 496 (C.C.P.A. 1970) (emphasis added).

In addition, even if all elements of a claim are disclosed in various prior art references, which is certainly not the case here as discussed below, the claimed invention taken as a whole still cannot be said to be obvious without some reason why one of ordinary skill at the time of the invention would have been prompted to modify the teachings of a reference or combine the teachings of multiple references to arrive at the claimed invention.

The controlling case law, rules, and guidelines repeatedly warn against using an applicant's disclosure as a blueprint to reconstruct the claimed invention. For example, the M.P.E.P. states, "The tendency to resort to 'hindsight' based upon applicant's disclosure is often difficult to avoid due to the very nature of the examination process. However, impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art." M.P.E.P. § 2142.

The U.S. Supreme Court's decision in *KSR Int'l Co. v. Teleflex, Inc.* reiterated the requirement that Examiners provide an explanation as to why the claimed invention would have been obvious. *KSR Int'l Co. v. Teleflex, Inc.*, 127 S.Ct. 1727 (2007). The analysis regarding an apparent reason to combine the known elements in the fashion claimed in the patent at issue "should be made explicit." *KSR*, 127 S.Ct. at 1740-41. "Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *Id.* at 1741 (internal quotations omitted).

The new examination guidelines issued by the PTO in response to the *KSR* decision further emphasize the importance of an explicit, articulated reason why the claimed invention is obvious. Those guidelines state, in part, that "[t]he key to supporting any rejection under 35 U.S.C. 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious. The Supreme Court in *KSR* noted that the analysis supporting a rejection under 35 U.S.C. 103 should be made explicit." *Examination Guidelines for Determining Obviousness Under 35 U.S.C. 103 in View of the Supreme Court Decision in KSR International Co. v. Teleflex Inc.*, 72 Fed. Reg. 57526, 57528-29 (Oct. 10, 2007) (internal citations omitted). The guidelines further describe a number of rationales that, in the PTO's view, can support a finding of obviousness. *Id.* at 57529-34. The guidelines set forth a number of particular findings of fact that must be made and explained by the Examiner to support a finding of obviousness based on one of those rationales. *See id.*

II. Claims 1-6, 8-13, and 15-20 are Allowable over the Proposed *Elmore-Gadbois-Moran* Combination

Independent Claim 1 of the present Application, as amended, recites:

> A method for use in a Web Services arrangement comprising:
> arranging User object(s) under a repository layer comprising one or more Repository objects collectively forming a Prefix, each User object representing a Web Services account;
> arranging Business Entity object(s) under User object(s);

arranging corresponding TModel object(s) under at least one of User object(s), Repository object and Prefix;

receiving a request to modify an object from a user;

matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object;

providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user; and

modifying the object as requested by the user in response to the user accessing the object.

This combination of features and operations is not disclosed, taught, or suggested in the proposed *Elmore-Gadbois-Moran* combination.

A. The proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "*matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object*"

As at least a first point of error, the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object," as recited in Claim 1. In the *Office Action*, the Examiner acknowledges that the recited operation is not disclosed in *Elmore* in view of *Gadbois*. (*Final Office Action*, page 3). It is the Examiner's position, however, that the additional disclosure of *Moran* discloses Appellants' "matching" step. (*Final Office Action*, page 3). Appellants respectfully disagree.

Moran discloses that "Access Control Lists (ACLs) are used to describe the permitted actions (permissions) on protected network computer system resources or objects associated with a client or user identity." (*Moran*, Abstract). Specifically, *Moran* discloses that an "Authorization Service (21) uses a central database that lists all resources in the secure domain and the ACL and POP policies assigned (attached) to each resource." (*Moran*, page 3, paragraph 60). According to *Moran*, "[t]his master authorization policy (31) database and the user registry (containing user and group accounts) are the key components that help define a network security policy." (*Moran*, page 3, paragraph 60). *Moran* further discloses that the "ACL policy is made up of one or more entries that include user specific permissions or rights." (*Moran*, page 3, paragraph 62). "ACL policies provide the Authorization Service with information to make a "yes" or "no" answer on a specific request to access a protected object, and to perform some operation on that object." (*Moran*, page 3, paragraph 62). An example ACL is shown in Figure 5 and merely includes "user and group designations, and their specific permissions." (*Moran*, page 4, paragraph 70). Thus, *Moran* merely relates to storing in a central database a list that identifies a user by user name and associates with the user name the objects that the user has permission to access.

For modification of an object by a user, *Moran* discloses:

> When a client or user initially requests permission to perform an action on a protected object, such as requesting to "read" or "modify" a protected system file, the client's ID is first authenticated by the authentication service. This results in a set of credentials being created, including the authenticated user's ID or user name, a list of user groups to which the user may belong, the name of the protected object being requested, and the action being requested. For example, the credentials and authorization request for Bill Smith of ABC Corporation . . . attempting to "modify" a protected system file "ABC_401k_summaries" may contain parameters such as:
>
> B_Smith, ABC_employees, "modify", "ABC_401k_summaries".

(*Moran*, page 3, paragraphs 46-47). After the creation of the credentials and authorization request, *Moran* discloses that the "user ID From the authenticated credentials is matched (91) with the requested protected object's ACL entries." (*Moran*, page 6, paragraph 115). Thus, according to the disclosure of *Moran*, the Authorization Service receives a user request identifying a user by a user name and a requested object. Then, "the userID from the authenticated credentials is matched (91) with the requested protected object's ACL entries." (*Moran*, page 6, paragraph 115). Thus, the Authorization Service looks up the user name in the ACL, which is stored in a centralized database. "The permissions granted (93) are those in the matching entry or entries." (*Moran*, page 6, paragraph 115).

As such, the "matching" referred to in *Moran* merely includes using a list to associate an object with a user to give that user permission to access the object. Appellants respectfully submit that accessing a list that includes a user name and a user object to which that user has access is not analogous to "matching a distinguished name associated with the user and **at least a portion of the distinguished name associated with the object**," as recited in Claim 1. There is no disclosure in *Moran* that one would be able to match the name of the user with at least a portion of the name of the object. In fact, and as discussed above, *Moran* actually discloses the credentials and authorization request for Bill Smith of

ABC Corporation . . . attempting to "modify" a protected system file "ABC_401k_summaries" merely include "B_Smith, ABC_employees, "modify", "ABC_401k_summaries". Thus, the name of the user is "Bill Smith" or "B_Smith" and the name of the file is "ABC_401k_summaries." The name of the file "ABC_401k_summaries" does not include the name of the user and cannot be matched with "B_Smith." Accordingly, *Moran* and the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "matching a distinguished name associated with the user and **at least a portion of the distinguished name associated with the object**," as recited in Claim 1.

B. The proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest *"providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user"*

As at least a second point of error, the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user," as recited in Claim 1. Again, the *Final Office Action* acknowledges that the recited operation is not disclosed in *Elmore* in view of *Gadbois* and instead relies upon the additional disclosure of *Moran.* (*Final Office Action*, page 3). For reasons similar to those discussed above in Section II(A) of this Appeal Brief, Appellants respectfully disagree.

As discussed above, *Moran* discloses that "Access Control Lists (ACLs) are used to describe the permitted actions (permissions) on protected network computer system resources or objects associated with a client or user identity." (*Moran*, Abstract). Specifically, *Moran* discloses that an "ACL policy is made up of one or more entries that include user specific permissions or rights" and is used to "provide the Authorization Service with information to make a "yes" or "no" answer on a specific request to access a protected object, and to perform some operation on that object." (*Moran*, page 3, paragraph 62). Thus, according to *Moran*, a list that identifies a user by user name and associates with the user name the objects that the user has permission to access is stored in a central database. As stated above, however, there is no disclosure in *Moran* that one would be able to match the name of the user with at least a portion of the name of the object. In fact, and as discussed above, *Moran* actually discloses the credentials and authorization request for Bill Smith of ABC

Corporation . . . attempting to "modify" a protected system file "ABC_401k_summaries" merely include "B_Smith, ABC_employees, "modify", "ABC_401k_summaries". Thus, the name of the user is "Bill Smith" or "B_Smith" and the name of the file is "ABC_401k_summaries." The name of the file "ABC_401k_summaries" does not include the name of the user and cannot be matched with "B_Smith." Because *Moran* and the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest matching a distinguished name associated with the user and **at least a portion of the distinguished name associated with the object**, the proposed combination also cannot be said to disclose, teach, or suggest "providing the user access to the object **in response to matching the distinguished name associated with the object and the distinguished name associated with the user**," as recited in Claim 1.

C. The Proposed *Elmore-Gadbois-Moran* Combination is Improper

Appellants respectfully submit that the proposed *Elmore-Gadbois-Moran* combination is improper, as applied to Claims 1-6, 8-13, 15-17 and 19. Specifically, Appellants submit that one of ordinary skill in the art at the time of Appellants' invention would not have been motivated to modify or combine the cited references in the manner the Examiner proposes. Appellants respectfully submit that these rejections are, therefore, improper and should be reversed by the Board.

The question raised under 35 U.S.C. § 103 is whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art at the time of the invention. Accordingly, even if all elements of a claim are disclosed in various prior art references, which is certainly not the case here as discussed above, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill at the time of the invention would have been prompted to modify the teachings of a reference or combine the teachings of multiple references to arrive at the claimed invention.

In this case, the Examiner has not provided the requisite teaching, suggestion, or motivation, either in the cited references or in the knowledge generally available to one of ordinary skill in the art at the time of Appellants' invention to modify or combine *Elmore* with the disclosures of *Gadbois* and *Moran*. As the basis for the proposed combination, the Examiner summarily concludes that "[i]t would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified *Elmore* by the teaching of *Gadbois*" to include Appellants' claim limitations because such a system "would enable an efficient means of recording and publishing assertions regarding business organization relationships (peer-to-peer, parent subsidiary, etc.) by different business organizations or their authorized publishing entities and efficient means of managing publisher assertions (*Gadbois*, paragraph 4). Additionally, the Examiner summarily concludes that "[i]t would have been obvious to a person having ordinary skill in the art at the time the invention was made to have further modified Elmore by the teaching of *Moran*" to include Appellants' claim limitations because such a system "would enable the use of access control lists to describe permissions on protected systems and networks to access resources and objects associated with clients, thus providing a more secure, protected system." (*Final Office Action*, pages 4-5). Thus, it appears that the Examiner has merely proposed alleged advantages of modifying *Elmore* to include features of *Gadbois* and *Moran* (advantages which Appellants do not admit could even be achieved by combining these references in the manner the Examiner proposes). While the Examiner has cited respective portions of the references that tout an alleged advantage of the disclosures, the cited portions do not provide an explanation as to: (1) why it would have been obvious to one of ordinary skill in the art at the time of Appellants' invention (without using Appellants' claims as a guide) to modify the particular techniques disclosed in *Elmore* with the cited disclosures of *Gadbois* and *Moran*; (2) how one of ordinary skill in the art at the time of Appellants' invention would have actually done so; and (3) how doing so would purportedly meet the limitations of Appellants' claims. Indeed, if it were sufficient for Examiners to merely point to a purported advantage of one reference and conclude that it would have been obvious to modify other references with that reference simply based on that advantage, then virtually any two or more references would be combinable just based on the fact that one reference states an advantage of its system. Of course, as the Federal Circuit has made clear and as discussed above, that is not the law.

For at least these reasons, Appellants submit that the rejection of Appellants' claims over the proposed *Elmore-Gadbois-Moran* combination is improper.

D. Conclusion

For these reasons, Appellants respectfully submit that Claim 1, together with Claims 2-6 and 17-18 that depend on Claim 1, are allowable over the proposed *Elmore-Gadbois-Moran* combination. Similar to Claim 1, independent Claim 8 recites "code for matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object" and "code for providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user" and "code for modifying the object as requested by the user in response to the user accessing the object." As such, for at least those reasons discussed above with regard to Claim 1, Claim 8, together with Claims 9-16 and 19-20 that depend on Claim 8, is also patentably distinguishable from and allowable over the proposed *Elmore-Gadbois-Moran* combination.

III. Claims 7 and 14 are Allowable over the Proposed *Elmore-Gadbois-Moran* Combination

Dependent Claims 7 and 14 depend upon independent Claims 1 and 8, which Appellants have shown above to be allowable. Accordingly, dependent Claims 7 and 17 are not obvious over the proposed *Elmore-Gadbois-Moran* combination at least because Claims 7 and 14 include the limitations of their respective independent claims. Dependent Claims 7 and 14 are additionally allowable because the claims recite claim elements that further distinguish the art.

A. The proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "*providing a Distinguished Name of an object revealing a chain of ownership and control for that object*"

As at least one additional point of error, the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "providing a Distinguished Name of an object revealing a chain of ownership and control for the object," as recited in Claim 7. In the *Final Office Action*, the Examiner acknowledges that the recited claim elements are not disclosed in *Elmore* and instead relies specifically upon the additional disclosure of *Gadbois* for the recited operation. (*Final Office Action*, page 6). Appellants respectfully disagree.

Gadbois discloses that within a DIT, "[a] first tier or set of nodes coupled to the host node include a set of nodes representative of organizations." (*Gadbois*, page 3, paragraph 27). Beneath the organization nodes, the DIT includes "a number of interior sub-nodes which contain further information, or links to further information, regarding the respective organization." (*Gadbois*, page 3, paragraph 28). As illustrated in Figure 2, these sub-nodes include "Groups," "Business Services," and "Publisher Assertions" associated with the Organization. However, while *Gadbois* discloses that nodes are organized by organization there is no disclosure in *Gadbois* that a name of an object reveals the chain of ownership and control for the object. For example, there is no disclosure that the Distinguished Names of each of "Groups," "Business Services," or "Publisher Assertions" specifically reveal the ownership and control for those objects. In fact, *Gadbois* is silent as to the naming of the of the sub-nodes. Accordingly, *Gadbois* and the proposed *Elmore-Gadbois-Moran* combination does not disclose, teach, or suggest "providing a Distinguished Name of an object revealing a chain of ownership and control for that object," as recited in Claim 7.

B. The Proposed *Elmore-Gadbois-Moran* Combination is Improper

Appellants respectfully submit that the proposed *Elmore-Gadbois-Moran* combination is improper as applied to Claim 7. Specifically, Appellants submit that one of ordinary skill in the art at the time of Appellants' invention would not have been motivated to modify or combine the cited references in the manner the Examiner proposes. Appellants respectfully submit that these rejections are, therefore, improper and should be reversed by the Board.

The question raised under 35 U.S.C. § 103 is whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art at the time of the invention. Accordingly, even if all elements of a claim are disclosed in various prior art references, which is certainly not the case here as discussed above, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill at the time of the invention would have been prompted to modify the teachings of a reference or combine the teachings of multiple references to arrive at the claimed invention.

In this case, the Examiner has not provided the requisite teaching, suggestion, or motivation, either in the cited references or in the knowledge generally available to one of ordinary skill in the art at the time of Appellants' invention to modify or combine *Elmore* with the disclosures of *Gadbois* and *Moran*. As stated above, the Examiner summarily concludes that "[i]t would have been obvious" to modify *Elmore* to include features disclosed in *Gadbois* because such a system "would enable an efficient means of recording and publishing assertions regarding business organization relationships (peer-to-peer, parent subsidiary, etc.) by different business organizations or their authorized publishing entities and efficient means of managing publisher assertions (*Gadbois*, paragraph 4). Likewise, the Examiner summarily concludes that "[i]t would have been obvious" to have further modified *Elmore* to include features disclosed in *Moran* because such a system "would enable the use of access control lists to describe permissions on protected systems and networks to access resources and objects associated with clients, thus providing a more secure, protected system." (*Final Office Action*, pages 4-5).

Again, it appears that the Examiner has merely proposed alleged advantages of modifying *Elmore* to include features of *Gadbois* and *Moran* (advantages which Appellants do not admit could even be achieved by combining these references in the manner the Examiner proposes). While the Examiner has cited respective portions of the references that tout an alleged advantage of the disclosures, the cited portions do not provide an explanation as to: (1) why it would have been obvious to one of ordinary skill in the art at the time of Appellants' invention (without using Appellants' claims as a guide) to modify the particular techniques disclosed in *Elmore* with the cited disclosures of *Gadbois* and *Moran*; (2) how one of ordinary skill in the art at the time of Appellants' invention would have actually done so; and (3) how doing so would purportedly meet the limitations of Appellants' claims. Indeed, if it were sufficient for Examiners to merely point to a purported advantage of one reference and conclude that it would have been obvious to modify other references with that reference simply based on that advantage, then virtually any two or more references would be combinable just based on the fact that one reference states an advantage of its system. Of course, as the Federal Circuit has made clear and as discussed above, that is not the law.

For at least these reasons, Appellants submit that the rejection of Appellants' claims over the proposed *Elmore-Gadbois-Moran* combination is improper.

C. Conclusion

For these reasons, Appellants respectfully submit that Claim 7 is allowable over the proposed *Elmore-Gadbois-Moran* combination. Similar to Claim 7, dependent Claim 14 recites "code for providing a Distinguished name of an object revealing a chain of ownership and control for the object." As such, for at least those reasons discussed above with regard to Claim 7, Claim 14 is also patentably distinguishable from and allowable over the proposed *Elmore-Gadbois-Moran* combination.

CONCLUSION

Appellants have demonstrated that the present invention, as claimed, is clearly distinguishable over the prior art cited by the Examiner. Therefore, Appellants respectfully request the Board to reverse the final rejections and instruct the Examiner to issue a Notice of Allowance with respect to all pending claims.

The Commissioner is hereby authorized to charge $540.00 for filing this Brief in support of an Appeal to Deposit Account No. 02-0384 of Baker Botts, L.L.P. No other fees are believed due; however, the Commissioner is authorized to charge any additional fees or credits to Deposit Account No. 02-0384 of Baker Botts, L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellants

Jenni R. Moen

Jenni R. Moen
Reg. No. 52,038
(214) 953-6809

Dated: December 2, 2008

Correspondence Address:

at Customer No. **05073**

APPENDIX A

Pending Claims

1. A method for use in a Web Services arrangement comprising:

arranging User object(s) under a repository layer comprising one or more Repository objects collectively forming a Prefix, each User object representing a Web Services account;

arranging Business Entity object(s) under User object(s);

arranging corresponding TModel object(s) under at least one of User object(s), Repository object and Prefix;

receiving a request to modify an object from a user;

matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object;

providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user; and

modifying the object as requested by the user in response to the user accessing the object.

2. The method as recited in claim 1, further comprising:

arranging Publisher Assertion object(s) under Business Entity object(s).

3. The method as recited in claim 1, further comprising:

providing Service Projection object(s) under Business Entity object(s).

4. The method as recited in claim 3, wherein the Service Projection object(s) is implemented as an alias.

5. The method as recited in claim 4, further comprising providing first field(s) as attributes of Publisher Assertion object(s).

6. The method as recited in claim 5, further comprising representing a keyed reference by an auxiliary class.

7. The method as recited in claim 6, further comprising providing a Distinguished Name of an object revealing a chain of ownership and control for the object.

8. A computer recording medium including computer executable code for use in a Web Services arrangement comprising:

code for arranging User object(s) under a repository layer comprising one or more Repository objects collectively forming a Prefix, each User object representing a Web Services account;

code for arranging Business Entity object(s) under User object(s);

code for arranging corresponding TModel object(s) under at least one of User object(s), Repository object and Prefix;

code for receiving a request to modify an object from a user;

code for matching a distinguished name associated with the user and at least a portion of the distinguished name associated with the object;

code for providing the user access to the object in response to matching the distinguished name associated with the object and the distinguished name associated with the user; and

code for modifying the object as requested by the user in response to the user accessing the object.

9. The computer recording medium as recited in claim 8, further comprising:

code for arranging Publisher Assertion object(s) under Business Entity object(s).

10. The computer recording medium as recited in claim 8, further comprising: providing Service Projection object(s) under Business Entity object(s).

11 . The computer recording medium as recited in claim 10, wherein the Service Projection object(s) is implemented as an alias.

12. The computer recording medium as recited in claim 11, further comprising code for providing first field(s) as attributes of Publisher Assertion object(s).

13. The computer recording medium as recited in claim 12, further comprising code for representing a keyed reference by an auxiliary class.

14. The computer recording medium as recited in claim 13, further comprising code for providing a Distinguished Name of an object revealing a chain of ownership and control for the object.

15. The method as recited in Claim 1, further comprising storing the arrangement of User object(s), one or more Repository objects, Business Entity object(s), and TModel object(s) in a registry accessible to one or more users of Web Services.

16. The computer recording medium as recited in Claim 8, further comprising code for storing the arrangement of User object(s), one or more Repository objects, Business Entity object(s), and TModel object(s) in a registry accessible to one or more users of Web Services.

17. The method as recited in Claim 1, further comprising:

providing a plurality of repository layers distributed on a plurality of servers, each repository layer comprising at least on repository object; and

assigning a domain name to each of the plurality of repository layers; and

wherein arranging User object(s) under a repository layer comprises arranging User object(s) under each of the repository objects.

18. The method as recited in Claim 1, further comprising:

providing a plurality of repository layers distributed on a plurality of servers, each repository layer comprising at least on repository object; and

logically representing each User object, Business Entity object, and TModel object on each server, each User object, Business Entity object, and TModel object only stored on a selected one of the plurality of servers.

19. The computer recording medium as recited in Claim 8, further comprising:

code for providing a plurality of repository layers distributed on a plurality of servers, each repository layer comprising at least on repository object; and

code for assigning a domain name to each of the plurality of repository layers; and

wherein the code for arranging User object(s) under a repository layer comprises code for arranging User object(s) under each of the repository objects.

20. The computer recording medium as recited in Claim 8, further comprising:

code for providing a plurality of repository layers distributed on a plurality of servers, each repository layer comprising at least on repository object; and

code for logically representing each User object, Business Entity object, and TModel object on each server, each User object, Business Entity object, and TModel object only stored on a selected one of the plurality of servers.

APPENDIX B

Evidence Appendix

No evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, and no other evidence was entered by the Examiner and relied upon by Appellants in the Appeal.

<u>APPENDIX C</u>

Related Proceedings Appendix

As stated on Page 3 of this Appeal Brief, to the knowledge of Appellants' Counsel, there are no known appeals, interferences, or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.